Dynamics Generalisation with Behaviour Foundation Models

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement learning agents perform poorly when faced with unseen dynamics. Recent work on Behaviour Foundation Models (BFMs) has produced agents capable of solving many unseen *tasks* in an environment assuming consistency between the dynamics described by the pre-training dataset and the testing environment. In this preliminary work, we relax this assumption and ask: can BFMs return performant policies for tasks in environments with different dynamics to that seen during training? We build on work that compensates for differences in dynamics by modifying the reward function the agent is trained against. We show that if the BFM's policy is prompted correctly, we can elicit behaviour required to solve a specific set of dynamics generalisation problems. We report some preliminary experiments on the ExORL benchmark and discuss next steps.

1 Introduction

Reinforcement Learning (RL) agents [43] struggle to adapt to novel contexts. Recent work on Behaviour Foundation Models (BFMs) has shown agents can be pre-trained to solve many unseen tasks in an environment, subject to an assumed consistency between the dynamics of the pre-training dataset and testing environment [47, 38, 35, 25]. In this preliminary work, we relax this assumption and ask: can BFMs return policies for tasks in environments with different dynamics to that seen during training?

Our response to this question builds on past work that compensates for differences in dynamics by modifying the reward function the agent is trained against [15]. The key idea is that because a BFM is pre-trained to return a policy for *any* reward function, it should have a policy that is performant under a range of changed dynamics if these changed dynamics can be codified as reward functions. Following [15], we find performant policies on unseen dynamics by updating the rewards used to prompt the BFM's policy such that trajectories that are possible and high reward in the test domain are elicited (Figure 1). We demonstrate the effectiveness of our approach with some preliminary experiments and discuss future work that will further test the method.

2 Preliminaries

Contextual markov decision processes. A Contextual Markov Decision Process (CMDP) is defined by $\{C, S, A, R, \rho, \gamma, \mathcal{M}(c)\}$. C is the set of contexts, S and A are sets of states and actions, $R : S \times A \to \mathbb{R}$ is a reward-function, γ is a discount factor, and ρ is the initial state distribution [20]. M is a function that maps a context $c \in C$ to a Markov Decision Process (MDP) $\mathcal{M}(c) = \{S, A, R, \rho, \gamma, T^c\}$ with a context-dependent transition function $T^c : S \times A \times C \to S$. Given $(s_0, a_0) \in S \times A$, a context $c \in C$, and a policy $\pi : S \to \Delta(A)$, we denote $\Pr(\cdot|s_0, a_0, \pi, c)$ and $\mathbb{E}(\cdot|s_0, a_0, \pi, c)$ the probabilities and expectations under state-actions sequences $(s_t, a_t)_{t\geq 0}$ starting at (s_0, a_0) and following policy π, with $s_t \sim T^c(\cdot|s_{t-1}, a_{t-1}, c)$ and $a_t \sim \pi(\cdot|s_t)$. We denote $\pi^*_{c,R}$ the optimal policy in context c for reward function R, which is the policy that maximises the expected discounted future reward i.e. $\pi^*_{c,R} = \arg\max_\pi \mathbb{E}(\gamma^t R(s_t, a_t)|s_0, a_0, \pi, c)$.

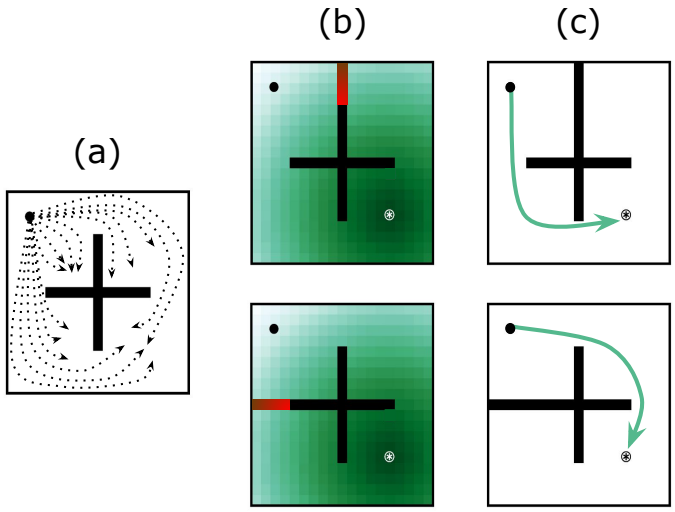

Figure 1: **Contextual behaviour foundation models.** (a) A BFM is pre-trained to solve any downstream task in a source environment. (b) An exploratory policy collects data in the test-time dynamics, and a binary classifier is trained to infer transitions that are not possible in the source environment and relabels them with low reward. (c) The BFMs policy is conditioned on the relabelled rewards to return a policy with high-reward on the test-time dynamics.

Behaviour foundation models. BFMs use unsupervised RL to pre-train an adaptive policy to solve any downstream task in an environment. Different BFM backbones have been explored with most leveraging successor measures [6, 46, 47, 25] or successor features [4, 7, 35]. We use a *forward-backward (FB)* BFM as our backbone because of its strong empirical zero-shot RL performance, though for this work the type of BFM we use is unimportant as discussed in Section 3. We recall the pertinent FB properties below, and refer the reader to [46] for a detailed explanation of its training protocol.

FB approximates the successor measures of near-optimal policies for any task. The successor measure $M^\pi(s_0, a_0, \cdot)$ over S is the cumulative discounted time spent in each future state s_{t+1} after starting in state s_0, taking action a_0, and following policy π thereafter:

$$M^\pi(s_0, a_0, X) := \sum_{t \geq 0} \gamma^t \Pr(s_{t+1} \in X | s_0, a_0, \pi) \; \forall \; X \subset \mathcal{S}. \tag{1}$$

Let ρ be an arbitrary state distribution, and $\mathbb{R}^d$ be a representation space. FB representations are composed of a *forward* model $F : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}^d$, a *backward* model $B : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, and set of polices $(\pi_z)_{z \in \mathbb{R}^d}$. They are trained such that

$$\begin{cases} M^{\pi_z}(s_0, a_0, X) \approx \int_X F(s_0, a_0, z)^\top B(s, a) \rho(\mathrm{d}s) & \forall \; s_0 \in \mathcal{S}, a_0 \in \mathcal{A}, X \subset \mathcal{S}, z \in \mathbb{R}^d, \\ \pi_z(s) \approx \arg\max_a F(s, a, z)^\top z & \forall \; (s, a) \in \mathcal{S} \times \mathcal{A}, z \in \mathbb{R}^d. \end{cases} \tag{2}$$

The task vector z for some downstream task r_{test} is inferred with a small number of reward-labelled states:

$$z_{\text{test}} \approx \mathbb{E}_{(s,a) \sim \mathcal{D}_{\text{labelled}}}[r_{\text{test}}(s) B(s)], \tag{3}$$

and passed as an argument to π_z. If z_{test} lies within the task sampling distribution $\mathcal{Z}$ used during pre-training, then $\pi_z(s) \approx \arg\max_a Q^{\pi_z}_{r_{\text{test}}}(s, a)$, and hence this policy is approximately optimal for r_{test}.

Problem 1 (Fast dynamics generalisation) We are interested in pre-training an agent in one context such that it is able to generalise to unseen contexts in the CMDP. We select one context for training $C_{\text{train}} = \{c_{\text{train}} \in C\}$, and use all other contexts for testing i.e. $C_{\text{test}} = \{c \in C | c \neq c_{\text{train}}\}$. For pre-training, the agent has access to a dataset of transitions $D = \{(s_i, a_i, r_i, s_{i+1})^{c_{\text{train}}}\}_{i=1}^{|D|}$ generated by an unknown, but highly exploratory, behaviour policy in the training context. At test time, the agent is provided a smaller dataset of reward-free transitions from each test context $d_c = \{(s_i, a_i, s_{i+1})^c\}_{i=1}^{|d|} \ \forall \ c \in C_{\text{test}}$. The agent must use these transitions to adapt its policy to the test contexts. Ideally, the agent should maximise the expected discounted return across all test contexts $\sum_{c \in C_{\text{test}}} \mathbb{E}[\sum_{t \geq 0} \gamma^t R(s_t, a_t) | s_0, a_0, \pi_c, c]$.

3 Method

To solve Problem 1 we will make two assumptions. The first follows previous work on zero-shot dynamics generalisation [15].

Assumption 1 *Assume every transition with non-zero probability across C_{test} will have non-zero probability in the training context:*

$$T^c(s_{t+1}|s_t, a_t) > 0 \Rightarrow T^{c_{\text{train}}}(s_{t+1}|s_t, a_t) > 0 \qquad \forall \ \ s_t, s_{t+1} \in S, a_t \in A, c \in C_{\text{test}}, \tag{4}$$

Intuitively, Assumption 1 says a wider range of trajectories/behaviours are possible in training context than in any of the test contexts. This is like a training facility where a soccer player can practice free-kicks from any position on the pitch, before being tasked with a free-kick from a specific position on match-day. If Assumption 1 did not hold, then the optimal policy for a test context may involve behaviour that is not possible in the training context, so it is unclear how a near-optimal policy could be derived from the training context alone. Our second assumption relates to the optimality of our pre-trained adaptive policy.

Assumption 2 *For some context $c \in C_{test}$, reward function R, and pre-trained adaptive policy π_z, assume there exists some z such that $\pi_z \approx \pi^*_{c,R}$.*

For Assumption 2 to hold, the trajectory followed by $\pi^*_{c,R}$ must exist in D, and our BFM must be optimised for R by sampling $z = \mathbb{E}_{(s,a) \sim d_c}[R(s,a)B(s,a)]$ from $\mathcal{Z}$ during pretraining. The former is satisfied if Assumption 1 holds and the pre-training dataset is sufficiently diverse i.e. it has been collected by an exploratory behaviour policy. If the latter does not hold, the dimensionality of the representation space $\mathbb{R}^d$ can be increased until it does.

Dynamics generalisation using rewards. By Assumption 2 our BFM is capable of returning a near-optimal policy at test-time if it's policy is conditioned on the correct z. Recall that FB infers z via a reward-weighted average of state-action pairs (Equation 3). Intuitively, one can think of Equation 3 as defining a goal-reaching task, where the goal is the state-action pair (s_g, a_g) with highest expected reward under R. If we were to naively infer z from the reward-labelled states in $d_{c_{\text{train}}}$, the agent may attempt to reach (s_g, a_g) by traversing states that are inaccessible under $T^{c_{\text{test}}}$. Our task is therefore to provide the agent with an augmented set of reward-labelled states with high reward for transitions possible under $T^{c_{\text{test}}}$ and low reward otherwise.

Inferring the reward augmentation with classifiers. The above idea is closely related to those explored in [15], and instantiated in their algorithm *Domain Adaption with Reward from Classifiers (DARC)*. They learn a policy in a training context which receives both high reward and has high likelihood under the changed dynamics in the test context. Let $p(\tau)$ be the desired distribution over trajectories in the test context, and $q(\tau)$ be the distribution over trajectories in the training context, then they codify the above objective as minimising the reverse KL divergence between the two distributions:

$$\min_{\pi(a|s)} D_{\text{KL}}(q \ || \ p) = -\mathbb{E}_{p_{\text{train}}}\big[\sum_t r(s_t, a_t) + \Delta r(s_t, a_t, s_{t+1})\big], \tag{5}$$

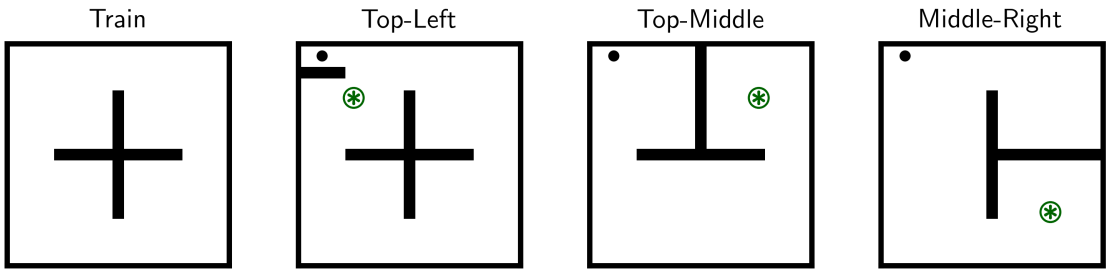

Figure 2: **Point mass-maze with changed dynamics.** TRAIN is the unmodified version of point mass-maze from the ExORL benchmark; we use the RND dataset to pre-train our BFMs. TOP-LEFT, TOP-MIDDLE, and MIDDLE introduce an obstacle that change the environment dynamics. The agent is initialised at • and tasked with reaching ⊛.

where $\Delta r(s_t, a_t, s_{t+1}) = \log p(s_{t+1}|s_t, a_t) - \log q(s_{t+1}|s_t, a_t)$ is a reward correction which penalises transitions with high likelihood in the training context, but not in the test context. In practice Δr is estimated with two binary classifiers such that

$$\Delta r(s_t, a_t, s_{t+1}) = \log p(\text{test}|s_t, a_t, s_{t+1}) - \log p(\text{test}|s_t, a_t) \\ - \log p(\text{train}|s_t, a_t, s_{t+1}) - \log p(\text{train}|s_t, a_t) \quad (6)$$

where red terms are the difference in logits from the classifier conditioned on (s_t, a_t, s_{t+1}) and the blue terms are the difference in the logits from the classifier conditioned on (s_t, a_t).

We can use the same protocol for amending the rewards used to infer task z. Provided the dataset of transitions from a test context $d_{c_{\text{test}}}$ and from the train context D, we train two binary classifiers to predict whether the state-action (s_t, a_t) and (s_t, a_t, s_{t+1}) came from the train context or test context. Then we relabel state-actions in D by adding Δr from Equation 6. z is this inferred using this augmented dataset of rewards via Equation 3. We call the FB model that uses augmented rewards to infer z for test dynamics: *Contextual* FB (CFB). Implementation details are provided in Appendix 2.

4 Preliminary Experiments

Setup. We create a modified version of the point-mass maze environment from the ExORL benchmark [51]–Figure 2. BFMs are pre-trained with the RND dataset [8] from unmodified dynamics, and evaluated on three test mazes with unseen obstacles that change the environment dynamics. We rollout an RND agent in each of these modified environments to create the dynamics inference datasets. More detail on the experimental protocol is provided in Appendix A.

Baselines. We use vanilla FB (i.e. without reward augmentation) as our BFM baseline, and Offline TD7 [18] as our oracle. FB does not see data from test environments and so rollouts the pre-trained policy for the task naive to any changes in the environment dynamics. TD7 is trained directly on modified environment datasets until convergence, and so represents the max performance one could expect to extract from the modified dynamics dataset.

Results. We report the aggregate performance of CFB and our baselines in Table 1. On the training context, all methods perform well on all tasks, though FB and CFB perform worse on `reach bottom right` (the hardest of the three targets) than the rest. On the test contexts, FB's performance drops significantly as its policy attempts to use gaps in parts of the maze that are now blocked by obstacles. CFB returns a policy that avoids the obstacle and receives higher reward. The methods never perform as well as the oracle, but this is inline with previous work that evaluates BFMs on this benchmark [47, 25].

Table 1: **Performance on Point-mass maze under changed dynamics.** Scores are the IQM of 10 rollouts where ± captures the 95% confidence interval. The max achievable score is 1000. NB: FB and CFB are identical in the train context because $\Delta r = 0$ for all transitions.

Context	Task	Obstacle	FB	CFB	Oracle (TD7)
Train	`reach top left`	-	930 ± 10	930 ± 10	985 ± 7
	`reach top right`	-	810 ± 33	810 ± 10	905 ± 12
	`reach bottom right`	-	405 ± 33	405 ± 33	901 ± 10
Test	`reach top left`	Top-Left	250 ± 31	890± 10	965 ± 11
	`reach top right`	Top-Middle	112 ± 5	676 ± 28	812 ± 24
	`reach bottom right`	Middle-Right	88 ± 13	395 ± 42	889 ± 18

Next Steps. In follow-up work we plan to evaluate the performance of CFB on locomotion tasks from the ExORL benchmark i.e. walker, cheetah and quadruped. We plan on creating environments analogous to the *crippled* MuJoCo tasks explored in several other works on dynamics generalisation [15, 33, 42]. We will test whether a pre-trained BFM can provide policies that generalise to robots with inhibited joint movement at test time. We also plan to test how sensitive CFB's performance is to the size of dataset recovered from the test contexts.

5 Related Work

Dynamics generalisation in RL. Dynamics generalisation in RL is a well-established problem [28, 32]. Common remedies include: data augmentation [39, 3, 50, 22, 21, 29], domain randomisation [45, 14, 26, 27, 36], learning context-aware policies [42, 30, 5], and meta-learning [11, 40, 16, 33, 34]. Our work is most similar to DARC [15] which augments rewards to tackle dynamics generalisation. Where DARC is concerned with generalising to one unseen test context, our method may be able to generalise to *any* unseen context as long as Assumptions 1 and 2 are satisfied.

Behaviour foundation models. The BFM machinery builds upon successor representations [13], universal value function approximators [41], successor features [4] and successor measures [6]. Modern methods use these ideas in the context of universal successor features (USF) [7] or forward-backward (FB) representations [46, 47, 25]. The USF features can be learned with diversity-based methods [31, 23], laplacian eigenfunctions [48], inverse curiosity modules [37], or contrastive learning [12]. Some works train RL foundation models with Transformers [10, 9, 19, 24, 40, 49, 52], but these methods do not have robust mechanism for conditioning on unseen tasks at test time. [38] use BFMs to perform fast imitation learning. This is a related idea, but, as with any imitation learning method, they require access to expert trajectories at test-time where we do not.

6 Conclusions

In this preliminary work we asked: can BFMs return policies for tasks in environments with different dynamics to that seen during training? We respond to this question by building on past work that compensates for differences in dynamics by modifying the reward function. Using two binary classifiers, we augment the rewards in the BFM's pre-training dataset so as to up-weight transitions that are likely under the test-time dynamics and down-weight those that are unlikely. In preliminary experiments on Point-mass maze from the ExORL benchmark, we showed that this reward-modification is enough to elicit performant behaviour from the BFM on a specific set of dynamics generalisation problems. We propose some future experiments to further test the method.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In *International Conference on Machine Learning*, pp. 619–629. PMLR, 2021.

[4] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[5] Michael Beukman, Devon Jarvis, Richard Klein, Steven James, and Benjamin Rosman. Dynamics generalisation in reinforcement learning via adaptive context-aware policies. *Advances in Neural Information Processing Systems*, 36, 2024.

[6] Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021.

[7] Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado Van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

[8] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[9] Yevgen Chebotar, Quan Vuong, Alex Irpan, Karol Hausman, Fei Xia, Yao Lu, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. *arXiv preprint arXiv:2309.10150*, 2023.

[10] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[11] Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

[13] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

[14] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

[15] Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *ICLR*, 2021.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

[17] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

[18] Scott Fujimoto, Wei-Di Chang, Edward Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: State-action representation learning for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[19] Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. *arXiv preprint arXiv:2111.10364*, 2021.

[20] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes, 2015.

[21] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13611–13617. IEEE, 2021.

[22] Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. *Advances in neural information processing systems*, 34:3680–3693, 2021.

[23] Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019.

[24] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[25] Scott Jeen, Tom Bewley, and Jonathan M. Cullen. Zero-shot reinforcement learning from low quality data. *NeurIPS Generalisation in Planning Workshop*, 2023.

[26] Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021.

[27] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021.

[28] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76: 201–264, 2023.

[29] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

[30] Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 5757–5766. PMLR, 2020.

[31] Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pp. 6736–6747. PMLR, 2021.

[32] Ishita Mediratta, Qingfei You, Minqi Jiang, and Roberta Raileanu. A study of generalization in offline reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[33] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[34] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671*, 2018.

[35] Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. *ICML*, 2024.

[36] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, pp. 17473–17498. PMLR, 2022.

[37] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

[38] Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In *International Conference on Learning Representations*, 2024.

[39] Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

[40] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[41] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

[42] Younggyo Seo, Kimin Lee, Ignasi Clavera Gilaberte, Thanard Kurutach, Jinwoo Shin, and Pieter Abbeel. Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:12968–12979, 2020.

[43] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[44] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[45] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

[46] Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021.

[47] Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations*, 2023.

[48] Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

[49] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 24631–24645, 17–23 Jul 2022.

[50] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[51] Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

[52] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pp. 27042–27059. PMLR, 2022.

Appendices

A Experimental Protocol

We conduct our experiments in a modified version of the Point-mass maze environment from the ExORL benchmark [51] and DeepMind Control Suite [44].

A.1 Point-mass Maze

Point-mass Maze is a 2D maze with four rooms where the task is to move a point-mass to one of the rooms. The state and action spaces are 4 and 2-dimensional respectively; the state space consists of x, y positions and velocities of the mass, the action space is the x, y tilt angle. ExORL provides four reaching tasks `reach top left, reach top right, reach bottom left` and `reach bottom right.` The mass is always initialised in the top left and the reward is proportional to the distance from the goal. We modify the reward function to be dense and proportional to the distance from the goal. We create three modified versions of the environment, visualised in Figure 2 and described below.

Top Left An obstacle is placed in the top left quadrant of the maze near where the mass spawns. The target is `reach top left`.

Top Middle The maze cross is extended to close the entrance to the top right quadrant from the top left quadrant. The target is `reach top right`.

Middle Right The maze cross is extended to close the entrance to the bottom right quadrant from the top right quadrant. The target is `reach bottom right`.

A.2 Datasets

We train our BFM using the RND dataset [8] from the ExORL benchmark. For Top Left, Top Middle, and Middle Right we create our on exploratory datasets by training an RND agent for 5 million timesteps and saving all observed transitions. We uniformly sub-sample 10,000 transitions from these to create our dynamics inference datasets for each test context. The state-occupancies are visualised in Figure 3.

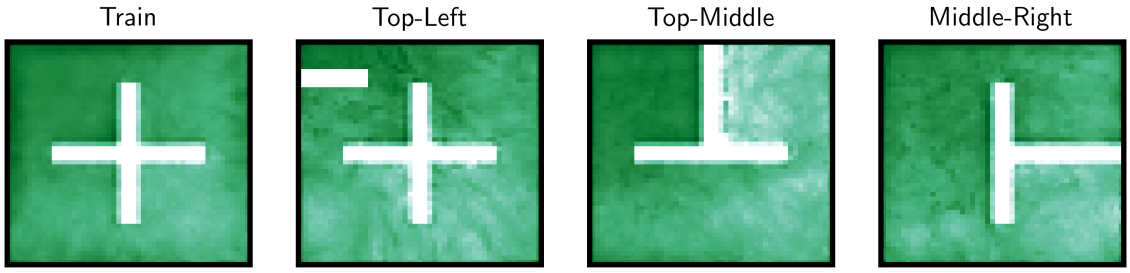

Figure 3: **Point-mass Maze datasets.** Train is the original RND dataset from the ExORL benchmark collected on unmodified dynamics. Top-Left, Top-Middle, and Top-Right are datasets we curated by running RND for 5 million steps in each modified environment.

A.3 Training and Evaluation Protocol

Training. FB and CFB are pre-trained for 1 million learning steps using the ExORL RND dataset. TD7 is trained for 1 million steps on the test context

Table 2: **Context Classifier Hyperparameters.**

Hyperparameter	Value
Hidden dimension	256
Hidden layers	2
Learning rate	3e-4
Batch size	256
Training dataset size	1e6
Learning steps	1e4

Evaluation. We evaluate the cumulative reward achieved by CFB and our baselines on each task across 5 seeds. We report scores as per the best practice recommendations of [1] by taking the IQM of 10 rollouts on the test task per seed, and averaging across seeds.

B Implementation Details

B.1 Context Classifiers

Following [15], we implement a residual parameterisation of the transition classifier. Using $f_{\text{trans}}(s_t, a_t, s_{t+1}), f_{\text{sa}}(s_t, a_t) \in \mathbb{R}^2$ to denote the outputs of the classifiers, the predictions are computed as:

$$q_{\text{sa}}(\cdot|s_t, a_t) = \text{SoftMax}(f_{\text{sa}}(s_t, a_t)) \tag{7}$$

$$q_{\text{trans}}(\cdot|s_t, a_t, s_{t+1}) = \text{SoftMax}(f_{\text{trans}}(s_t, a_t, s_{t+1}) + f_{\text{sa}}(s_t, a_t)). \tag{8}$$

We apply Gaussian noise with $\sigma = 1$ to the inputs of both classifiers. Hyperparameters are reported in Table 2. They are trained to minimise the standard cross-entropy loss:

$$\mathcal{L}_{\text{trans}} = -\mathbb{E}_{\mathcal{D}_{\text{test}}}[\log q_{\text{trans}}(\text{test}|s_t, a_t, s_{t+1})] - \mathbb{E}_{\mathcal{D}_{\text{train}}}[\log q_{\text{trans}}(\text{train}|s_t, a_t, s_{t+1})] \tag{9}$$

$$\mathcal{L}_{\text{sa}} = -\mathbb{E}_{\mathcal{D}_{\text{test}}}[\log q_{\text{sa}}(\text{test}|s_t, a_t)] - \mathbb{E}_{\mathcal{D}_{\text{train}}}[\log q_{\text{sa}}(\text{train}|s_t, a_t)] \tag{10}$$

B.2 Forward-Backward Representations

B.2.1 Architecture

The forward-backward architecture described below follows the implementation by [47] exactly, other than the batch size which we reduce from 1024 to 512. We did this to reduce the computational expense of each run without limiting performance. The hyperparameter study in Appendix J of [47] shows this choice is unlikely to affect FB performance. All other hyperparameters are reported in Table 3.

Forward Representation $F(s, a, z)$. The input to the forward representation F is always preprocessed. State-action pairs (s, a) and state-task pairs (s, z) have their own preprocessors P_F^1 and P_F^2. P_F^1 and P_F^2 are feedforward MLPs that embed their inputs into a 512-dimensional space. These embeddings are concatenated and passed through a third feedforward MLP F which outputs a d-dimensional embedding vector.

Backward Representation $B(s)$**.** The backward representation B is a feedforward MLP that takes a state as input and outputs a d-dimensional embedding vector.

Actor $\pi(s, z)$**.** Like the forward representation, the inputs to the policy network are similarly preprocessed. State-action pairs (s, a) and state-task pairs (s, z) have their own preprocessors P_π^1

Table 3: **FB Hyperparameters.**

Hyperparameter	Value
Latent dimension d	50 (100 for maze)
F hidden layers	2
F hidden dimension	1024
B hidden layers	3
B hidden dimension	256
P_F hidden layers	2
P_F hidden dimension	1024
P_π hidden layers	2
P_π hidden dimension	1024
Std. deviation for policy smoothing σ	0.2
Truncation level for policy smoothing	0.3
Learning steps	1,000,000
Batch size	512
Optimiser	Adam
Learning rate	0.0001
Discount γ	0.98 (0.99 for maze)
Activations (unless otherwise stated)	ReLU
Target network Polyak smoothing coefficient	0.01
z-inference labels	10,000
z mixing ratio	0.5

and P_π^2. P_π^1 and P_π^2 are feedforward MLPs that embed their inputs into a 512-dimensional space. These embeddings are concatenated and passed through a third feedforward MLP which outputs a a-dimensional vector, where a is the action-space dimensionality. A Tanh activation is used on the last layer to normalise their scale. As per [17]'s recommendations, the policy is smoothed by adding Gaussian noise σ to the actions during training.

Misc. Layer normalisation [2] and Tanh activations are used in the first layer of all MLPs to standardise the inputs.

B.2.2 z Sampling

FB representations require a method for sampling the task vector z at each learning step. [47] employ a mix of two methods, which we replicate:

1. Uniform sampling of z on the hypersphere surface of radius $\sqrt{d}$ around the origin of $\mathbb{R}^d$,
2. Biased sampling of z by passing states $s \sim \mathcal{D}$ through the backward representation $z = B(s)$. This also yields vectors on the hypersphere surface due to the $L2$ normalisation described above, but the distribution is non-uniform.

We sample z 50:50 from these methods at each learning step.

B.3 TD7

We adopt the original implementation and hyperparameters from `https://github.com/sfujim/TD7` `commit c1c280d`. Hyperparameters are reported in Table 4.

Critic(s). TD7 employs double Q networks. The critics are feedforward MLPs that take a state-action pair (s, a) as input and output a value $\in \mathbb{R}^1$.

Table 4: **TD7 Hyperparameters.**

Hyperparameter	**Value**
Hidden layers (for all networks)	2
Hidden dimension (for all networks)	256
Latent dimension z	256
Target policy noise σ	$\mathcal{N}(0, 0.2^2)$
Target policy noise clipping c	(-0.5, 0.5)
Policy update frequency	2
Probability smoothing α	0.4
Minimum priority	1
Behaviour cloning weight λ	0.1
Checkpoint criteria	minimum
Early assessment episodes	1
Late assessment episodes	20
Early time steps	750k
Criteria reset weight	0.9
Early time steps	750k
Criteria reset weight	0.9
Optimiser	Adam
Learning rate	3e-4
Discount γ	0.99
Seed steps	25k
Activations (unless otherwise stated)	ELU
Batch size	256
Target update frequency	250
Exploration noise	$\mathcal{N}(0, 0.1^2)$

Actor. The actor is a standard feedforward MLP taking the state s as input and outputting an a-dimensional vector, where a is the action-space dimensionality. The policy is smoothed by adding Gaussian noise σ to the actions during training.

Encoders. TD7 has a state encoder f and a state-action encoder g. f takes the state outputs a latent state $z \in \mathbb{R}^2 56$. g takes as input the latent state z and action a and outputs a second latent embedding.